Bipartite Perfect Matching Benchmarks

Cayden R. Codel, Joseph E. Reeves, Marijn J. H. Heule, and Randal E. Bryant
Carnegie Mellon University, Pittsburgh, United States

Intro

The pigeonhole and mutilated chessboard problems are challenging benchmarks for most SAT solvers not employing special reasoning techniques. The solvers that do employ special techniques can efficiently solve the canonical versions of these two problems, but may fail with even slight problem variations. To evaluate and improve the robustness of SAT solvers, we designed a benchmark family of perfect matching problems on bipartite graphs that generalizes the pigeonhole and mutilated chessboard problems [1]. Our benchmark generator supports various encodings and randomized constructions. These formulas are generally hard in the absence of reasoning techniques resilient under randomness and encoding variations.

Bipartite Problem and Encoding

Random bipartite graphs are used to explore non-structured problem instances of the perfect matching problem. The *density* of a bipartite graph with node partitions of size n and m is defined as the ratio of the number of edges to the number of possible edges $n \times m$. To generate a random connected bipartite graph, edges are added randomly to a random spanning tree until the desired density is reached.

Given a connected bipartite graph, a Boolean variable is associated with each edge such that a satisfying assignment is the edges in a perfect matching. The problem is encoded as a CNF with at-least-one (ALO) constraints on nodes from the larger partition and at-most-one (AMO) constraints on nodes in the smaller partition. This is the *Sparse* problem encoding. The *Full* problem encoding is derived by using both ALO and AMO constraints for each node.

AMO constraints are encoded in three ways: Pairwise, Sinz, and Linear,

Pairwise$(x_1, ..., x_n)$ is the pairwise set of binary clauses with no auxiliary variables:

$(\overline{x}_i \vee \overline{x}_j)$ with $1 \leq i < j \leq n$

Sinz$(x_1, ..., x_n)$ introduces signal variables that propagate the AMO condition:

$$\overline{x}_i \vee s_i \quad \text{for } 1 \leq i \leq n \qquad \overline{s}_i \vee s_{i+1}, \ \overline{s}_i \vee \overline{x}_{i+1} \quad \text{for } 1 \leq i < n$$

Linear$(x_1, ..., x_n)$ introduces variables to split up the Pairwise encoding when $n > 4$:

$$\text{Pairwise}(x_1, x_2, x_3, y) \wedge \text{AMO}(\overline{y}, x_4, .., x_n)$$

The **Mixed** AMO constraint option selects one of the three AMO encodings at random for each node independently. Note the signal s_n could be left out for the Sinz encoding of AMO, and is in our implementation.

Bounded Variable Elimination on Pigeonhole

Experimental findings [2] revealed a performance decline for top-tier solvers when bounded variable elimination (BVE) [3] was enabled on pigeonhole formulas. To explore this phenomenon, we started with a pigeonhole formula using the Sparse problem encoding and Pairwise AMO encoding, then applied BVE to some set of variables and gave solvers the new formula to solve. We found that specific variable elimination orderings generated formulas that are difficult for all solvers tested. Namely, eliminating n variables coming from independent pigeons and independent holes. Notably, this elimination ordering is forced in the Full problem encoding for solvers that employ BVE.

Benchmarks

We submitted 21 benchmarks to the 2021 SAT Competition. The first 17 formulas represent three configurations for random bipartite problem generation: (1) Sparse with Pairwise AMO, (2) Sparse with Mixed AMO (denoted by Mix in the naming), and (3) Full (denoted by B in the naming) with Mixed AMO. For each configuration we construct iteratively larger graphs with partition sizes n from $15..20$, with the exception of the first configuration starting at $n = 16$. Each graph has edges added until a density of 0.5 is reached which is generally hard as seen in 1.

4 formulas are pigeonhole formulas with n from $11..14$ and the Sparse with Pairwise AMO encoding. BVE is applied to n variables for each (denoted by e# in the naming), with eliminated variables selected from independent pigeons and independent holes. This elimination order proves difficult for previous competition winners shown in 2.

All formulas are UNSAT.

References

[1] C. Codel, J. Reeves, M. Heule, and R. Bryant, "Bipartite perfect matching benchmarks," in *Proceedings of Pragmatics of (SAT)*, 2021.

[2] J. Reeves and M. Heule, "The impact of bounded variable elimination on solving pigeonhole formulas," in *Proceedings of Pragmatics of (SAT)*, 2021.

[3] N. Eén and A. Biere, "Effective preprocessing in SAT through variable and clause elimination," in *Theory and Applications of Satisfiability Testing (SAT)*, ser. LNCS, vol. 3569. Springer, 2005, pp. 61–75.

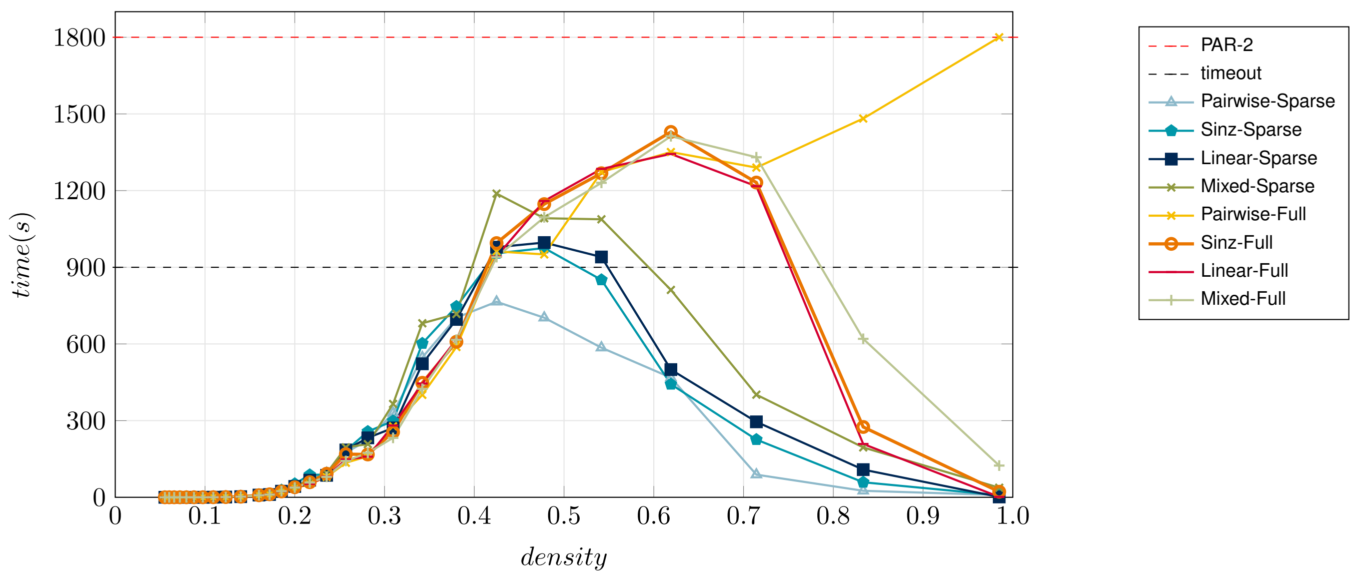

Fig. 1. Average execution time for KISSAT (2020 version) over randomly generated bipartite matching problems using a 900 second timeout and 1800 second PAR-2 score. For each density value, 60 random bipartite graphs are generated with a fixed edge count (130). Experiments cover the different AMO and problem encodings described above. Problems are harder around the 0.5 density, and the Mixed AMO encodings are difficult for the respective Sparse and Full encodings.

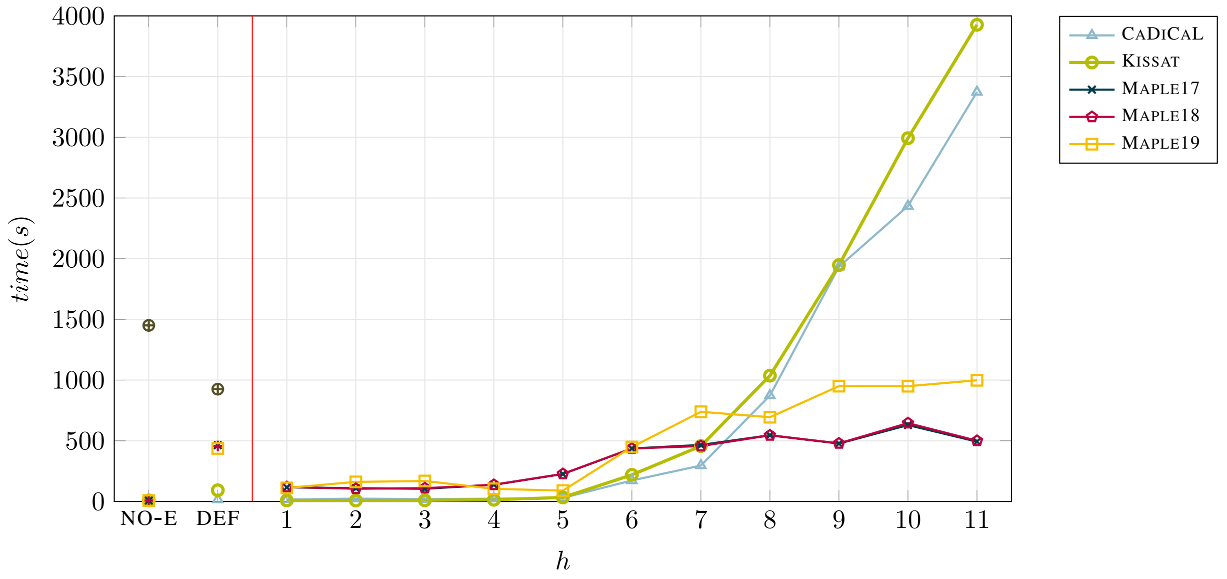

Fig. 2. Execution time on BVE instances of pigeonhole formula $n = 11$. In each h instance (x-axis) 12 variables are eliminated, selected from independent pigeons and h independent holes. NO-E is the solvers on pigeonhole $n = 11$ with BVE disabled, and DEF is the default configuration on the pigeonhole formula. Solvers are previous SAT competition winners. $h = 11$ represents the independent pigeon/hole BVE instance used in the $n = 11$ benchmark, and this variable elimination ordering is extended to $n = 12, 13, 14$. It is the most difficult formula for the solvers, though some experience larger performance degradation.